ENGINEERING PROGRAMMING I

Chapter 08

2017

Practice 1: Leap Year

- Write a C++ program in which the computer prompts the user to enter a four-digit year.
- When users enter a year, the user should be told if the year is a leap year or not.
- The algorithm is as follows.

Divide the year by 4.
If the remainder isn't zero,
    The year is not a leap year
Otherwise divide the year by 100 and
If the remainder isn't 0,
    The year is a leap year
    Otherwise, divide the year by 400 and
    If the remainder isn't 0
        The year is not a leap year
        Otherwise, The year is a leap year

Practice 2: Number-Guessing Game

- Write a C++ program in which the computer selects a random number in the range of 0 to 2000, and players get a maximum of 10 attempts to guess it.
- In each attempt, players should be told whether the guessed number is greater or smaller than the hidden number. If they match, players should be told they won.
- At the end of each game, players should be asked whether they want to play again.
- When the user quits, the program should output the total number of wins and losses.

- Hand in your hand-written code.

Random Number Generator

- **`rand`**
  - Returns a random number between **`0`** and the largest **`int`** the computer holds
  - Will yield same sequence of numbers each time the program is run
- **`srand(x)`**
  - Initializes random number generator with **`unsigned int x`**
  - Should be called at most once in a program

- These require **`cstdlib`** header file

Mathematical Library Functions

- These require **`cmath`** header file
- Take **`double`** arguments and return a **`double`**
- Commonly used functions

`abs`	Absolute value
`sin`	Sine
`cos`	Cosine
`tan`	Tangent
`sqrt`	Square root
`log`	Natural (e) log

Chapter 8: Arrays

Starting Out with C++
Early Objects
Seventh Edition

by Tony Gaddis, Judy Walters,
and Godfrey Muganda

Addison-Wesley
is an imprint of
PEARSON

Topics

- 8.1 Arrays Hold Multiple Values
- 8.2 Accessing Array Elements
- 8.3 Inputting and Displaying Array Contents
- 8.4 Array Initialization
- 8.5 Processing Array Contents
- 8.6 Using Parallel Arrays
- 8.7 The typedef Statement
- 8.8 Arrays as Function Arguments
- 8.9 Two-Dimensional Arrays
- 8.10 Arrays with Three or More Dimensions
- 8.11 Vectors
- 8.12 Arrays of Class Objects

Arrays Hold Multiple Values

- Array: variable that can store multiple values of the same type
- Values are stored in adjacent memory locations
- Declared using `[]` operator

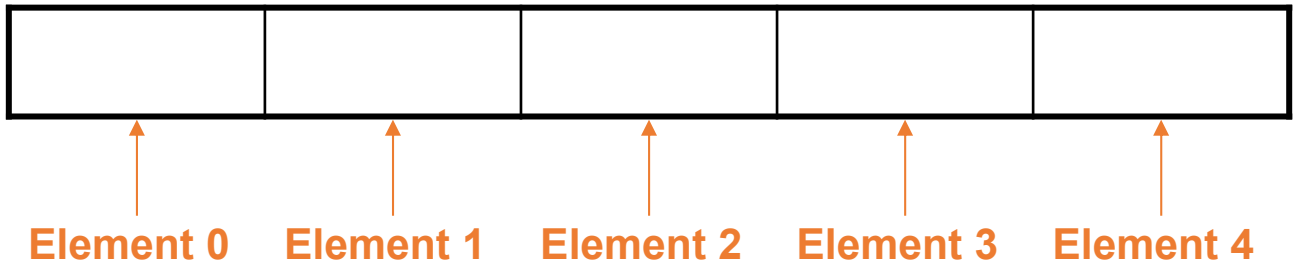
```
const int ISIZE = 5;
int tests[ISIZE];
```

Array Storage in Memory

The definition

```
int tests[ISIZE];  // ISIZE is 5
```

allocates the following memory



Array Terminology

In the definition `int tests[ISIZE];`

- **`int`** is the data type of the array elements
- **`tests`** is the name of the array
- **`ISIZE`**, in **`[ISIZE]`**, is the size declarator. It shows the number of elements in the array.
- The size of an array is the number of bytes allocated for it

  *(number of elements) * (bytes needed for each element)*

Array Terminology Examples

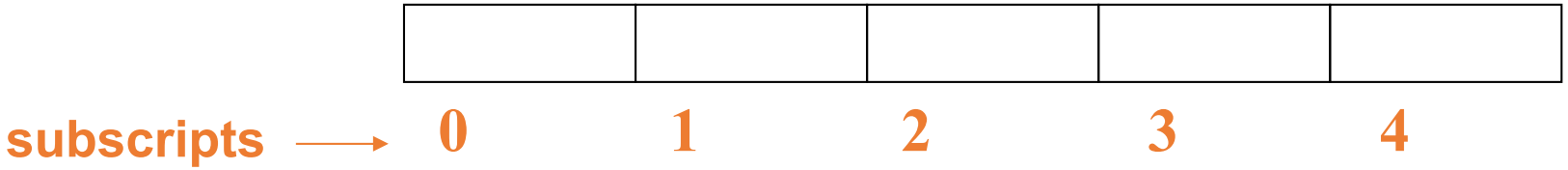
Examples:

Assumes **int** uses 4 bytes and **double** uses 8 bytes

```
const int ISIZE = 5, DSIZE = 10;
int tests[ISIZE]; // holds 5 ints, array
                  // occupies 20 bytes
double volumes[DSIZE];// holds 10 doubles
                      // array is 80 bytes
```

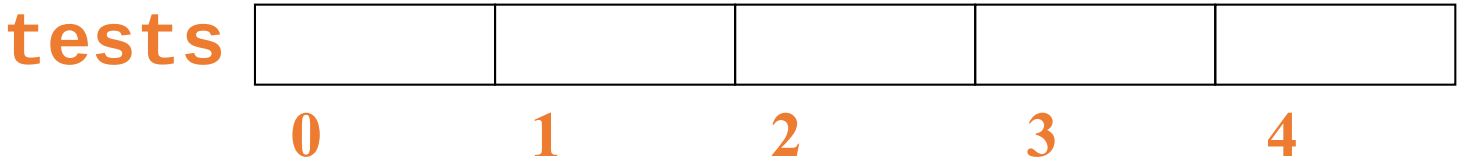
Accessing Array Elements

- Each array element has a subscript, used to access the element.
- Subscripts start at 0



Accessing Array Elements

Array elements (accessed by array name and subscript) can be used as regular variables




```
tests[0] = 79;
cout << tests[0];
cin  >> tests[1];
tests[4] = tests[0] + tests[1];
cout << tests; // illegal due to
               // missing subscript
```

Inputting and Displaying Array Contents

cout and **cin** can be used to display values from and store values into an array

```
const int ISIZE = 5;

int tests[ISIZE]; // Define 5-elt. array
cout << "Enter first test score ";
cin  >>  tests[0];
```

Array Subscripts

- Array subscript can be an integer constant, integer variable, or integer expression
- Examples:

	Subscript is
`cin >> tests[3];`	int constant
`cout << tests[i];`	int variable
`cout << tests[i+j];`	int expression

Inputting and Displaying All Array Elements

To access each element of an array

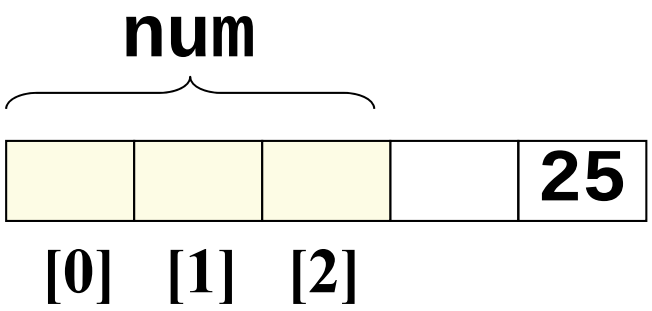
- Use a loop
- Let the loop control variable be the array subscript
- A different array element will be referenced each time through the loop

```
for (i = 0; i < 5; i++)
   cout << tests[i] << endl;
```

No Bounds Checking

- There are no checks in C++ that an array subscript is in range
- An invalid array subscript can cause program to overwrite other memory
- Example:

```
const int ISIZE = 3;
int i = 4;
int num[ISIZE];
num[i] = 25;
```



Off-By-One Errors

- Most often occur when a program accesses data one position beyond the end of an array, or misses the first or last element of an array.
- Don't confuse the ordinal number of an array element (first, second, third) with its subscript (0, 1, 2)

Array Initialization

- Can be initialized during program execution with assignment statements

```
tests[0] = 79;
tests[1] = 82; // etc.
```

- Can be initialized at array definition with an initialization list

```
const int ISIZE = 5;
int tests[ISIZE] = {79,82,91,77,84};
```

Start at element 0 or 1?

- May choose to declare arrays to be one larger than needed. This allows you to use the element with subscript 1 as the 'first' element, etc., and may minimize off-by-one errors.
- Element with subscript 0 is not used.
- This is most often done when working with ordered data, *e.g.*, months of the year or days of the week

Partial Array Initialization

- If array is initialized at definition with fewer values than the size declarator of the array, remaining elements will be set to **0** or **NULL**

```
int tests[ISIZE] = {79, 82};
```

79	82	0	0	0

- Initial values used in order; cannot skip over elements to initialize noncontiguous range

Implicit Array Sizing

- Can determine array size by the size of the initialization list

```
short quizzes[]={12,17,15,11};
```

- Must use either array size declarator or initialization list when array is defined

Processing Array Contents

- Array elements can be
  - treated as ordinary variables of the same type as the array
  - used in arithmetic operations, in relational expressions, etc.
- Example:

```
if (principalAmt[3] >= 10000)
  interest = principalAmt[3] * intRate1;
else
  interest = principalAmt[3] * intRate2;
```

Using Increment and Decrement Operators with Array Elements

When using ++ and -- operators, don't confuse the element with the subscript

```
tests[i]++;  // adds 1 to tests[i]
tests[i++];  // increments i, but has
             // no effect on tests
```

Copying One Array to Another

- Cannot copy with an assignment statement:

```
tests2 = tests;  //won't work
```

- Must instead use a loop to copy element-by-element:

```
for (int indx=0; indx < ISIZE; indx++)
   tests2[indx] = tests[indx];
```

Are Two Arrays Equal?

- Like copying, cannot compare in a single expression:

```
if (tests2 == tests)
```

- Use a while loop with a boolean variable:

```
bool areEqual=true;
int indx=0;
while (areEqual && indx < ISIZE)
{
   if(tests[indx] != tests2[indx]
      areEqual = false;
}
```

Sum, Average of Array Elements

- Use a simple loop to add together array elements

```
float average, sum = 0;
for (int tnum=0; tnum< ISIZE; tnum++)
    sum += tests[tnum];
```

- Once summed, average can be computed

```
average = sum/ISIZE;
```

Largest Array Element

- Use a loop to examine each element and find the largest element (*i.e.*, one with the largest value)

```
int largest = tests[0];
for (int tnum = 1; tnum < ISIZE; tnum++)
{  if (tests[tnum] > largest)
       largest = tests[tnum];
}
cout << "Highest score is " << largest;
```

- A similar algorithm exists to find the smallest element

Partially-Filled Arrays

- The exact amount of data (and, therefore, array size) may not be known when a program is written.
- Programmer makes best estimate for maximum amount of data, sizes arrays accordingly. A sentinel value can be used to indicate end-of-data.
- Programmer must also keep track of how many array elements are actually used

C-Strings and string Objects

Can be processed using array name

- Entire string at once, or
- One element at a time by using a subscript

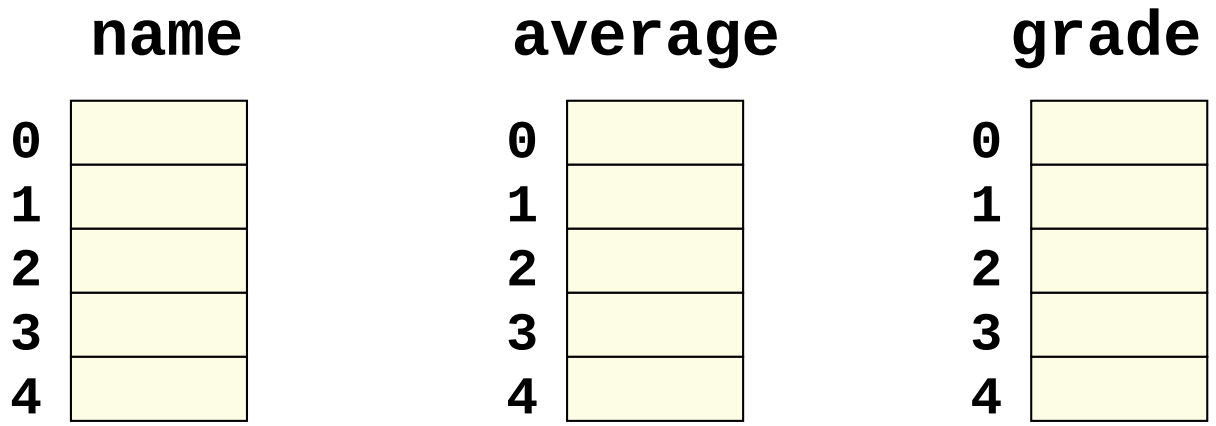
```
string city;
cout << "Enter city name: ";
cin  >> city;
```

Using Parallel Arrays

- Parallel arrays: two or more arrays that contain related data
- Subscript is used to relate arrays
  - elements at same subscript are related
- The arrays do not have to hold data of the same type

Parallel Array Example

```
const int ISIZE = 5;
string name[ISIZE];   // student name
float average[ISIZE]; // course average
char grade[ISIZE];    // course grade
```



Parallel Array Processing

```
const int ISIZE = 5;
 string name[ISIZE];   // student name
float average[ISIZE]; // course average
char grade[ISIZE];    // course grade
...
for (int i = 0; i < ISIZE; i++)
   cout << " Student: " << name[i]
        << " Average: " << average[i]
       << " Grade: "   << grade[i]
       << endl;
```